neously. As with other functionality, while it would be possible to design an API with explicit context; it would be less convenient to use. Further, by having workspaces support most of the same interfaces as the repository session object, programs written to the non-versioned API will work against workspaces with no code changes.

CONCLUSION

Maintaining versions and workspaces in an object has been described. As those of skill in the art will appreciate, the embodiments of the invention provide advantages not found in previous systems. For example, there is no need to copy objects as new versions of an object are created. The new versions are included in the range defined by the start version and end version identifiers. It is only when a property is actually updated that the property table representing the objects must be updated. Thus the embodiments of the invention make more efficient use of both memory and processor resources than previous systems.

Furthermore, the embodiments of the invention operate with both version-aware and non-version-aware applications.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention.

For example, those of ordinary skill within the art will appreciate that while maintaining versions and workspaces has been described in terms of an object database or repository, other means of storing persistent objects can be readily substituted. In addition, the embodiments of the invention have been described in terms of maintaining versions and workspaces associated with objects. However, the systems and methods described can be applied to any data entity serving a similar purpose to objects in an object-oriented environment. The terminology used in this application is meant to include all of these environments. Therefore, it is manifestly intended that this invention be limited only by the following claims and equivalents thereof.

We claim:

1. A computerized method for updating a version of an object having a property, the method comprising:

receiving an updated value for the property;

setting an end version field in a first data structure to a value representing a predecessor version of the object; and

creating a second data structure by:

setting a start version field in the second data structure to a value representing the successor version of the object;

setting an end version field in the second data structure object table to a value representing a most recent version of the object; and

setting a property value field in the second data structure to the updated value for the property, wherein the start version field and the end version field in the second data structure define a range of versions including the updated value for the property;

wherein version and property value fields of the data structures record properties of the object and associated versions of the object facilitating a recalling and generating of the object without requiring a copying of the object.

2. The computerized method of claim **1**, wherein the value representing the most recent version is infinity.

3. The computerized method of claim **1**, wherein the data structure is a row in a database.

4. The computerized method of claim **1**, wherein the object is a COM (Component Object Model) object.

5. The method of claim **1**, wherein a version of an object having a property is updated and recorded without copying the updated object.

6. The method of claim **1**, wherein a version of an object having a property is updated by copying only updated properties and associated version identifiers and not the updated object.

7. The method of claim **1**, wherein the property is a name-value pair and wherein the name refers to and performs operations on the value.

8. The method of claim **1**, wherein the method creates data structure only for properties that have changed value and does not copy the updated object.

9. The method of claim **1**, wherein the method creates or modifies property value fields only for properties that have changed value and creates or modifies version fields for only the versions including the properties that have changed value.

10. A computer-readable medium having computer-executable instructions for updating a version of an object having a property, the method comprising:

receiving an updated value for the property;

setting an end version field in a first data structure to a value representing a predecessor version of the object; and

creating a second data structure by:

setting a start version field in the second data structure to a value representing the successor version of the object;

setting an end version field in the second data structure to a value representing a most recent version of the object; and

setting a property value field in the second data structure to the updated value for the property, wherein the start version field and the end version field in the second data structure define a range of versions including the updated value for the property;

wherein version and property value fields of the data structures record properties of the object and associated versions of the object facilitating a recalling and generating of the object without requiring a copying of the object.

11. The computer-readable medium of claim **10**, wherein the value representing the most recent value is infinity.

12. The computer-readable medium of claim **10**, wherein the object is a COM (Component Object Model) object.

13. A method for propagating a relationship of a predecessor object to a successor object, said relationship having an origin object and a destination object, the method comprising:

reading a propagation flag on the relationship; and

if the propagation flag is set then performing the tasks of:

determining if a previously added version of the destination object has been added;

upon determining the previously added version has been added:

setting an end version field in a first data structure with a value representing a predecessor version of the object;